

High Speed Compressed Sensing Reconstruction in Dynamic Parallel MRI Using Augmented Lagrangian and Parallel Processing

Çağdaş Bilen, Yao Wang and Ivan Selesnick

Department of Electrical Engineering, Polytechnic Institute of NYU, Brooklyn, NY, USA

Abstract—Magnetic Resonance Imaging (MRI) is one of the fields that the compressed sensing theory is well utilized to reduce the scan time significantly leading to faster imaging or higher resolution images. It has been shown that a small fraction of the overall measurements are sufficient to reconstruct images with the combination of compressed sensing and parallel imaging. Various reconstruction algorithms has been proposed for compressed sensing, among which Augmented Lagrangian based methods have been shown to often perform better than others for many different applications. In this paper, we propose new Augmented Lagrangian based solutions to the compressed sensing reconstruction problem with analysis and synthesis prior formulations. We also propose a computational method which makes use of properties of the sampling pattern to significantly improve the speed of the reconstruction for the proposed algorithms in Cartesian sampled MRI. The proposed algorithms are shown to outperform earlier methods especially for the case of dynamic MRI for which the transfer function tends to be a very large matrix and significantly ill conditioned. It is also demonstrated that the proposed algorithm can be accelerated much further than other methods in case of a parallel implementation with graphics processing units (GPUs).

I. INTRODUCTION

Compressed sensing and sparse reconstruction methods have been popular topics of research especially in the last decade. Under certain conditions, such as the data being sparse, i.e. with a few non-zero coefficients in a domain that is incoherent with measurement domain, compressed sensing enables reliable recovery of signals even if they are measured at a rate below the Nyquist rate [1]. This stimulated research in many different fields in which data acquisition is limited due to constraints.

Medical imaging is one of the application areas that adopted compressed sensing principles rather quickly. It is shown that in magnetic resonance imaging (MRI), the number of measurement samples and thus the scan time can be reduced while preserving image quality if compressed sensing principles are used [2]. This is further improved by parallel imaging with algorithms such as SparseSENSE utilizing multiple coils [3]. These initial studies with compressed sensing were on still images or volumes and spatial total variation (TV) or wavelets were used as regularization constraints. Dynamic imaging is shown to benefit from compressed sensing even further due to the images being significantly correlated along temporal dimension and therefore represented with sparse temporal transforms. Similarly to the spatial case, temporal TV and wavelets are commonly used in MRI [4], [5]. The temporal

Fourier transform is also utilized with k-t SparseSENSE [6] especially with cardiac MRI, due to the cardiac motion being periodic therefore sparse with respect to Fourier transform. K-t Group Sparse SENSE is introduced by Usman et al. [7] which groups pixels with respect to their spatio temporal activity to treat static and dynamic regions differently during reconstruction.

Parallel computing has gained a great deal of interest in recent years and processors with multiple cores has become a standard in commercial computing. With the introduction of NVidia CUDA and ATI Stream architectures, it is possible to make use of large number of processor cores in graphics processing units (GPUs) for general purpose computing. Image reconstruction in the field of MRI has been shown to benefit from parallel processing and GPUs for real-time reconstruction [8], [9]. Regardless of the algorithm used, the reconstruction in compressed sensing MRI takes significantly more time compared to reconstruction of regular MRI. However real-time or fast reconstruction is essential in order for the commercial implementations of compressed sensing MRI to be useful in the field of medicine. Therefore fast reconstruction algorithms that can be implemented in parallel are necessary for compressed sensing MRI to be commercially viable.

In all the compressed sensing based approaches to medical imaging, the images are reconstructed by enforcing sparsity of the signal in a separate transform domain while simultaneously having the constraint of consistency with the measurements. This is often formulated as a basis pursuit problem and many different algorithms has been proposed for the solution such as iterative shrinkage based methods (FISTA [10], TwIST [11]). The Augmented Lagrangian based methods or mainly Alternating Direction Method of Multipliers (ADMM) approach [12] which is mostly equivalent to the Split-Bregman Method [13] has gained significant popularity especially in recent years due to its flexibility, ease of implementation and speed. SALSA algorithm which is also based on ADMM, has been shown to converge faster than other popular methods in various different imaging inverse problems [14] combining the ideas in ADMM with other denoising methods. Parallel MRI however suffers from slower implementations due to the transfer function being large as well as harder to adapt to ADMM formulation efficiently. Recently ADMM approach has been formulated for parallel MRI reconstruction [15], which mainly deals with still image reconstruction with spatial regularization, however the speed of the presented algorithms

has not been evaluated for dynamic MRI reconstruction with temporal regularization, which lead to very large and severely ill-conditioned transfer functions.

In this paper, we present two ADMM based solutions to analysis and synthesis prior formulations for compressed sensing dynamic parallel MRI reconstruction and propose an efficient implementation of the transfer function for the proposed ADMM algorithms in order to aid significantly faster reconstruction. The proposed method exploits the commonly used Cartesian subsampling pattern in MRI to provide faster and efficient implementation. It also benefits more from a parallel implementation such as in GPUs. It is demonstrated by the experiments that the proposed algorithm is faster and can be accelerated further than earlier methods when implemented with GPUs.

II. PROBLEM FORMULATION

A. Compressed Sensing and Parallel MRI

In magnetic resonance imaging, the measurements related to the image of a 2D slice of a volume are acquired in the spatial Fourier transform domain, which can be represented as

$$\begin{aligned} \mathbf{y}_t &= \mathbf{F}\mathbf{x}_t + \mathbf{n} \\ &= \mathbf{F}_h\mathbf{F}_v\mathbf{x}_t + \mathbf{n} \end{aligned} \quad (1)$$

where \mathbf{F}_h and \mathbf{F}_v are matrices representing Fourier transform operation along horizontal and vertical axes and \mathbf{F} represents the 2D spatial Fourier transform. \mathbf{x}_t and \mathbf{y}_t represent image of the 2D slice and corresponding Fourier measurements at time t respectively and \mathbf{n} represents the random noise in the measured samples that is modelled as Gaussian. The equation in (1) can easily be expanded for 3D volumes instead of slices, but all the derivations in this paper will consider 2D slices for simplicity. The vector \mathbf{x}_t is defined as

$$\mathbf{x}_t \triangleq \begin{bmatrix} \mathbf{x}_{v,1,t} \\ \vdots \\ \mathbf{x}_{v,n_h,t} \end{bmatrix}, \mathbf{x}_{v,i,t} \triangleq \begin{bmatrix} x_{1,i,t} \\ \vdots \\ x_{n_v,i,t} \end{bmatrix} \quad (3)$$

where $x_{j,i,t}$ is the image pixel at horizontal position i , vertical position j and at time t of an image of $n_h \times n_v$ spatial resolution and n_t frames.

Parallel MRI uses multiple receiving coils to speed up the data acquisition by making use of the redundancy among different coils. The measurement made by c -th coil can be represented as

$$\mathbf{y}_{t,c} = \mathbf{F}\mathbf{S}_c\mathbf{x}_t + \mathbf{n} \quad (4)$$

$$= \mathbf{F}_h\mathbf{F}_v(\mathbf{s}_c \odot \mathbf{x}_t) + \mathbf{n} \quad (5)$$

$$\mathbf{s}_c \triangleq \begin{bmatrix} \mathbf{s}_{v,1,c} \\ \vdots \\ \mathbf{s}_{v,n_h,c} \end{bmatrix}, \mathbf{s}_{v,i,c} \triangleq \begin{bmatrix} s_{1,i,c} \\ \vdots \\ s_{n_v,i,c} \end{bmatrix}$$

where $s_{j,i,c}$ indicates the sensitivity or weight associated with pixel $x_{j,i,t}$ by coil c . The matrix \mathbf{S}_c is a diagonal matrix that has the vector \mathbf{s}_c along its diagonal, while \odot represents element by element multiplication of two vectors. The sensitivity matrices are acquired after calibration and often noise decorrelation so that \mathbf{n} has a Gaussian distribution.

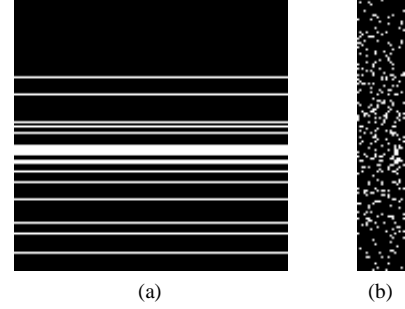


Fig. 1. A subsampling pattern example in dynamic MRI where only 1/8 of the samples are scanned (represented by white points) (a) Subsampling pattern for single frame in spatial Fourier domain (b) Sampled vertical indices along time in Dynamic MRI

In compressed sensing, only a subset of the measurements (often selected randomly) are acquired therefore reducing the required scan time in MRI. In Cartesian sampling, all the Fourier coefficients along the horizontal axis corresponding a specific vertical coordinate are measured sequentially and the samples cannot be skipped to save the acquisition time. Each new horizontal line however requires a switching of magnetic field gradient, which has a minimal switching time limit and the total time spent for acquisition is proportional to the number of scanned vertical lines, therefore the subsampling is usually only along vertical axis in the Fourier domain. In dynamic MRI, the subsampled vertical coordinates are also changed in time randomly. This process can be represented for each coil as

$$\tilde{\mathbf{y}}_{t,c} = \mathbf{H}_{t,c}\mathbf{x}_t + \mathbf{n} \quad (6)$$

$$= \mathbf{F}_h\mathbf{M}_{v,t}\mathbf{F}_v\mathbf{S}_c\mathbf{x}_t + \mathbf{n} \quad (7)$$

$$= \mathbf{F}_h\tilde{\mathbf{F}}_{v,t}\mathbf{S}_c\mathbf{x}_t + \mathbf{n} \quad (8)$$

$$\mathbf{H}_{t,c} \triangleq \mathbf{F}_h\tilde{\mathbf{F}}_{v,t}\mathbf{S}_c \quad (9)$$

where $\tilde{\mathbf{y}}_{t,c}$ is the vector of the sampled subset of measurements for coil c at time t and $\tilde{\mathbf{F}}_{v,t} = \mathbf{M}_{v,t}\mathbf{F}_v$ is the subsampled Fourier transform along the vertical axis which is composed of subsampled Fourier Transform matrices $\tilde{\mathbf{F}}_{v,i,t}$ transforming along each horizontal coordinate i

$$\tilde{\mathbf{F}}_{v,t} = \begin{bmatrix} \tilde{\mathbf{F}}_{v,1,t} & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & \tilde{\mathbf{F}}_{v,n_h,t} \end{bmatrix} \quad (10)$$

\mathbf{F}_h in (7) represents the Fourier transform along the horizontal axis similar to (5), but the size of the matrix might be different due to subsampling. Other subsampling patterns are also possible using different methods of scanning [2]. A sample subsampling pattern for dynamic MRI can be seen in Figure 1.

The transfer function in dynamic MRI with parallel coils can be represented as

$$\mathbf{y} = \tilde{\mathbf{F}}\mathbf{S}\mathbf{x} + \mathbf{n} \quad (11)$$

$$= \mathbf{H}\mathbf{x} + \mathbf{n} \quad (12)$$

where \mathbf{y} , \mathbf{x} , $\tilde{\mathbf{F}}$, \mathbf{S} and \mathbf{H} can be defined as

$$\mathbf{x} \triangleq \begin{bmatrix} \mathbf{x}_1 \\ \vdots \\ \mathbf{x}_{n_t} \end{bmatrix}, \mathbf{y} \triangleq \begin{bmatrix} \mathbf{y}_1 \\ \vdots \\ \mathbf{y}_{n_t} \end{bmatrix}, \mathbf{y}_t \triangleq \begin{bmatrix} \tilde{\mathbf{y}}_{t,1} \\ \vdots \\ \tilde{\mathbf{y}}_{t,n_c} \end{bmatrix} \quad (13)$$

$$\mathbf{H} \triangleq \begin{bmatrix} \mathbf{H}_1 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & \mathbf{H}_{n_t} \end{bmatrix}, \mathbf{H}_t \triangleq \begin{bmatrix} \mathbf{H}_{t,1} \\ \vdots \\ \mathbf{H}_{t,n_c} \end{bmatrix} \quad (14)$$

$$\mathbf{S} \triangleq \begin{bmatrix} \mathbf{S}^* & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & \mathbf{S}^* \end{bmatrix}, \mathbf{S}^* \triangleq \begin{bmatrix} \mathbf{S}_1 \\ \vdots \\ \mathbf{S}_{n_c} \end{bmatrix}, \quad (15)$$

$$\tilde{\mathbf{F}} \triangleq \begin{bmatrix} \mathbf{F}_1^* & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & \mathbf{F}_{n_t}^* \end{bmatrix}, \mathbf{F}_t^* \triangleq \begin{bmatrix} \mathbf{F}_h \tilde{\mathbf{F}}_{v,t} & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & \mathbf{F}_h \tilde{\mathbf{F}}_{v,t} \end{bmatrix} \quad (16)$$

and satisfy $\mathbf{H} = \tilde{\mathbf{F}}\mathbf{S}$. \mathbf{S} and \mathbf{F}_t^* are block diagonal matrices that are composed of n_t and n_c blocks along the diagonal respectively.

Compressed sensing theory shows that the signal \mathbf{x} can be recovered from \mathbf{y} almost certainly if,

- 1) $\Psi\mathbf{x}$ is sufficiently sparse for a known transform Ψ ($\Psi'\Psi = \Psi\Psi' = \mathbf{I}$, $'$ indicating the conjugate transpose)
- 2) \mathbf{H} and Ψ are sufficiently incoherent, i.e. the off diagonal entries of $\Psi\mathbf{H}'\mathbf{H}\Psi'$ after normalization are sufficiently small

One of the breakthroughs in compressed sensing is that the second condition is shown to be satisfied when \mathbf{H} is formed by random entries, or by a random subset of rows of a full rank matrix as in (6) regardless of Ψ [1], [2]. The first condition is also relaxed such that Ψ can be an overcomplete transform or dictionary or a penalty operator as in case of total variation [16]. Under these conditions it is shown that \mathbf{x} can be recovered almost certainly with the minimization

$$\tilde{\mathbf{w}} = \arg \min_{\mathbf{w}} \|\mathbf{w}\|_0 \text{ s.t. } \|\mathbf{y} - \mathbf{H}\Psi'\mathbf{w}\|_2^2 \leq \epsilon \quad (17)$$

$$\tilde{\mathbf{x}} = \Psi'\tilde{\mathbf{w}} \quad (18)$$

in which $\|\cdot\|_0$ is the L_0 -norm (which is in fact a quasi-norm) that is the number of non-zero entries of a vector ($\|\mathbf{x}\|_p \triangleq \left(\sum_i x_i^p\right)^{1/p}$ for $p > 0$). The constraint $\|\mathbf{y} - \mathbf{H}\Psi'\mathbf{w}\|_2^2 \leq \epsilon$ ensures consistency with the measurements and it is optimum for i.i.d. Gaussian noise with standard deviation ϵ . The minimization in (17) is non-convex and therefore practical methods do not guarantee convergence to global minimum. It is shown in the sparse reconstruction literature that the minimization of L_1 -norm as in

$$\tilde{\mathbf{w}} = \arg \min_{\mathbf{w}} \|\mathbf{w}\|_1 \text{ s.t. } \|\mathbf{y} - \mathbf{H}\Psi'\mathbf{w}\|_2^2 \leq \epsilon \quad (19)$$

$$\tilde{\mathbf{x}} = \Psi'\tilde{\mathbf{w}} \quad (20)$$

will lead to the same solution as in (17) provided that \mathbf{w} is sufficiently sparse [17]. The implications of this equivalence is significant since (19) is a convex optimization problem with

a global minimum and can be solved very efficiently with methods such as Alternating Direction Method of Multipliers (ADMM) [18] or the Split-Bregman Method [13]. In order to further simplify the minimization, the constraint in (19) can be removed to form the unconstrained formulation

$$\tilde{\mathbf{w}} = \arg \min_{\mathbf{w}} \lambda \|\mathbf{w}\|_1 + \|\mathbf{y} - \mathbf{H}\Psi'\mathbf{w}\|_2^2 \quad (21)$$

$$\tilde{\mathbf{x}} = \Psi'\tilde{\mathbf{w}} \quad (22)$$

A number of fast minimization algorithms exist for the solution of unconstrained minimization in (21) such as TwIST, FISTA and SALSA ([10], [14] and references therein) and the result is equivalent to (19) if λ is adjusted properly.

The formulation in (19) is referred to as synthesis prior formulation and (21) can be used with overcomplete transforms such as wavelets or non-orthogonal dictionaries. An alternative approach uses an analysis prior formulation with a penalty term for regularization as in

$$\tilde{\mathbf{x}} = \arg \min_{\mathbf{x}} \lambda R(\mathbf{x}) + \|\mathbf{y} - \mathbf{H}\mathbf{x}\|_2^2 \quad (23)$$

in which $R(\mathbf{x})$ is the penalty function that is large when \mathbf{x} has characteristics different from a prior knowledge of \mathbf{x} . A commonly used example for such penalty functions is the total variation (TV) which is defined for 1D signals as

$$\text{TV}(\mathbf{x}) \triangleq \sum_i |x_i - x_{i-1}| \quad (24)$$

and penalizes the signal gradient. In case of dynamic MRI, temporal regularization is often employed and TV can be defined along the temporal axis as

$$\begin{aligned} \text{TV}_t(\mathbf{x}) &= \sum_{t=2}^{n_t} \|\mathbf{x}_t - \mathbf{x}_{t-1}\|_1 \\ &= \left\| \begin{bmatrix} -\mathbf{I} & \mathbf{I} & 0 & \cdots & 0 \\ 0 & -\mathbf{I} & \mathbf{I} & \cdots & 0 \\ \vdots & & \ddots & & \vdots \\ 0 & \cdots & 0 & -\mathbf{I} & \mathbf{I} \end{bmatrix} \begin{bmatrix} \mathbf{x}_1 \\ \vdots \\ \mathbf{x}_{n_t} \end{bmatrix} \right\|_1 \\ &= \|\mathbf{D}_t \mathbf{x}\|_1 \end{aligned} \quad (25)$$

in which \mathbf{D}_t is the temporal difference matrix. Minimization of (23) with $R(\mathbf{x}) = \text{TV}_t(\mathbf{x})$ is possible with algorithms such as MFISTA [10] or ADMM based methods [13], [14].

B. ADMM Basics

Alternating Direction Method of Multipliers (ADMM) is fast convex minimization algorithm that makes use of variable splitting and Augmented Lagrangian to solve various forms of objective functions. For a general minimization problem such as

$$\arg \min_{\mathbf{z}} f(\mathbf{z}) = f_1(\mathbf{z}) + f_2(\mathbf{G}\mathbf{z}) \quad (26)$$

which may be non-trivial to directly solve for, ADMM algorithm solves the equivalent problem

$$\arg \min_{\mathbf{z}, \mathbf{v}} f(\mathbf{z}, \mathbf{v}) = f_1(\mathbf{z}) + f_2(\mathbf{v}) \quad (27)$$

such that $\mathbf{G}\mathbf{z} = \mathbf{v}$

Solving for (27) can be much easier than solving for (26) if functions f_1 , f_2 and the auxiliary variable \mathbf{v} are selected properly. The ADMM algorithm that solves for (27) can be summarized as iteratively applying

$$\mathbf{z} \leftarrow \arg \min_{\mathbf{z}} f_1(\mathbf{z}) + \mu \|\mathbf{Gz} - \mathbf{v} - \mathbf{d}\|_2^2 \quad (28)$$

$$\mathbf{v} \leftarrow \arg \min_{\mathbf{v}} f_2(\mathbf{v}) + \mu \|\mathbf{Gz} - \mathbf{v} - \mathbf{d}\|_2^2 \quad (29)$$

$$\mathbf{d} \leftarrow \mathbf{d} - (\mathbf{Gz} - \mathbf{v}) \quad (30)$$

until convergence after initializing \mathbf{v} , \mathbf{d} and μ [12]. The iterative steps (28)-(30) solving (27) is also known as the Split-Bregman Method, and is guaranteed to converge given $\mu > 0$, however convergence can be very slow unless μ is selected properly [13]. The parameter μ can be selected empirically for a given problem but more discussion on how to select μ theoretically can be found in [13], [12] and the references therein.

III. RELATED ALGORITHMS

The ADMM approach has been proposed to solve problems of the form (23) before, such as the Split-Bregman Method proposed in [13] minimizing

$$\tilde{\mathbf{x}} = \arg \min_{\mathbf{x}} \lambda \|\mathbf{v}\|_1 + \|\mathbf{y} - \mathbf{Hx}\|_2^2 \quad (31)$$

such that $\mathbf{v} = \mathbf{Rx}$

assuming the penalty function is in the form $R(\mathbf{x}) = \|\mathbf{Rx}\|_1$. This formulation is also proposed in [15] (named as P1) for Parallel MRI and can be solved iteratively with the steps similar to (28)-(30)

$$\begin{aligned} \mathbf{v} &\leftarrow \arg \min_{\mathbf{v}} \lambda \|\mathbf{v}\|_1 + \mu \|\mathbf{Rx} - \mathbf{v} - \mathbf{d}\|_2^2 \\ &= \mathfrak{S}(\mathbf{Rx} - \mathbf{d}, \frac{\lambda}{2\mu}) \end{aligned} \quad (32)$$

$$\begin{aligned} \mathbf{x} &\leftarrow \arg \min_{\mathbf{x}} \|\mathbf{y} - \mathbf{Hx}\|_2^2 + \mu \|\mathbf{Rx} - \mathbf{v} - \mathbf{d}\|_2^2 \\ &= (\mu \mathbf{R}'\mathbf{R} + \mathbf{H}'\mathbf{H})^{-1} [\mathbf{H}'\mathbf{y} + \mu \mathbf{R}'(\mathbf{v} + \mathbf{d})] \end{aligned} \quad (33)$$

$$\mathbf{d} \leftarrow \mathbf{d} - (\mathbf{Rx} - \mathbf{v}) \quad (34)$$

where $\mathfrak{S}(\cdot)$ is the element-wise soft thresholding function defined as

$$\mathfrak{S}(a, \tau) \triangleq \begin{cases} (|a| - \tau) \frac{a}{|a|} & \text{if } |a| > \tau \\ 0 & \text{if } |a| \leq \tau \end{cases} \quad (35)$$

Note that (33) requires $(\mu \mathbf{R}'\mathbf{R} + \mathbf{H}'\mathbf{H})^{-1}$ which is not invertible unless \mathbf{R} and \mathbf{H} have disjoint null spaces. In the case that it is invertible, a closed form solution is often not available especially if $\mathbf{R}'\mathbf{R} \neq \mathbf{I}$ and $\mathbf{H}'\mathbf{H} \neq \mathbf{I}$. For the Parallel MRI with a large \mathbf{H} , it is suggested in [15] that the term in (33) be calculated with few steps of Conjugate Gradient (CG) algorithm at each iteration of P1. However the use of CG iterations is not preferable due to being slow and not precise and can greatly slow down reconstruction when \mathbf{H} or \mathbf{R} is very large.

SALSA proposed in [14] uses an alternative approach and minimizes

$$\begin{aligned} \tilde{\mathbf{x}} &= \arg \min_{\mathbf{x}} \lambda \|\mathbf{Rv}\|_1 + \|\mathbf{y} - \mathbf{Hx}\|_2^2 \\ &\text{such that } \mathbf{v} = \mathbf{x} \end{aligned} \quad (36)$$

assuming an efficient method exists to solve

$$\mathbf{v} \leftarrow \arg \min_{\mathbf{v}} \lambda \|\mathbf{Rv}\|_1 + \mu \|\mathbf{v} - \mathbf{x} - \mathbf{d}\|_2^2 \quad (37)$$

which sometimes might be another iterative algorithm such as Chambolle's Method used for TV regularization [19]. SALSA replaces $(\mu \mathbf{R}'\mathbf{R} + \mathbf{H}'\mathbf{H})^{-1}$ in (33) with $(\mu \mathbf{I} + \mathbf{H}'\mathbf{H})^{-1}$ which is easier to compute and has a closed form solution for a variety of applications, but still often relies on iterative algorithms to solve for (37). Although SALSA has been shown to perform much better than popular methods such as FISTA, MFISTA or TwIST, it can be very slow if $(\mu \mathbf{I} + \mathbf{H}'\mathbf{H})^{-1}$ cannot be computed efficiently [14].

In addition to the algorithms summarized above which have been suggested for generic problems, a second algorithm is proposed in [15] (named as P2) specifically for Parallel MRI problem proposing more number of auxiliary variables such that

$$\begin{aligned} \tilde{\mathbf{x}} &= \arg \min_{\mathbf{x}} \lambda \|\mathbf{v}\|_1 + \|\mathbf{y} - \tilde{\mathbf{F}}\mathbf{p}\|_2^2 \\ &\text{such that } \mathbf{v} = \mathbf{Rm}, \mathbf{m} = \mathbf{x}, \mathbf{p} = \mathbf{Sx} \end{aligned} \quad (38)$$

where $\tilde{\mathbf{F}}$ and \mathbf{S} are subsampled Fourier transform and sensitivity matrices respectively as defined in (15) and (16) and satisfy $\mathbf{H} = \tilde{\mathbf{F}}\mathbf{S}$. The ADMM algorithm solving (38) requires calculation of the terms $(\mu \mathbf{I} + \mathbf{R}'\mathbf{R})^{-1}$, $(\mu \mathbf{I} + \tilde{\mathbf{F}}'\tilde{\mathbf{F}})^{-1}$ and $(\mu \mathbf{I} + \mathbf{S}'\mathbf{S})^{-1}$ each of which has a closed form solution and can be calculated fast and accurately at each iteration. However although each iteration of the algorithm solving (38) is quite fast, the number of iterations needed for convergence is much larger than (31) or (36) due to having too many auxiliary variables, and the reported speed is comparable to or worse than (31) with CG iterations [15].

IV. PROPOSED METHODS

Considering the drawbacks of the algorithms mentioned in Section III, we propose two algorithms to solve synthesis and analysis prior formulations respectively, which do not depend on any iterative steps except the iterations of ADMM and make use of the fast computation of $(\mu \mathbf{I} + \mathbf{H}'\mathbf{H})^{-1}$ proposed in Section IV-C.

A. ADMM for Synthesis Prior

Basic variable splitting approach in (27) can be used to solve (21) by

$$\begin{aligned} \tilde{\mathbf{w}} &= \arg \min_{\mathbf{w}} \lambda \|\mathbf{v}\|_1 + \|\mathbf{y} - \mathbf{H}\Psi'\mathbf{w}\|_2^2 \\ &\text{such that } \mathbf{v} = \mathbf{w} \end{aligned} \quad (39)$$

$$\tilde{\mathbf{x}} = \Psi'\tilde{\mathbf{w}} \quad (40)$$

which in turn can be minimized with iterations similar to (32)-(34) except \mathbf{x} is replaced with \mathbf{w} , \mathbf{R} with \mathbf{I} and \mathbf{H} with $\mathbf{H}\Psi'$.

Algorithm 1 ADMM minimization to solve (21)

```

1: procedure ADMM_Synthesis( $\mathbf{y}, \mathbf{H}, \Psi, \lambda$ )
   Minimizes  $\lambda\|\mathbf{v}\|_1 + \|\mathbf{y} - \mathbf{H}\Psi'\mathbf{w}\|_2^2$  s.t.  $\mathbf{w} = \mathbf{v}, \mathbf{x} = \Psi'\mathbf{w}$ 

2:   Set  $\mu > 0, \mathbf{d} = 0, \mathbf{w} = \Psi\mathbf{H}'\mathbf{y}$ 
3:   while convergence criteria not met do
4:      $\mathbf{v} \leftarrow \arg \min_{\mathbf{v}} \lambda\|\mathbf{v}\|_1 + \mu\|\mathbf{w} - \mathbf{v} - \mathbf{d}\|_2^2$ 
        $= \mathfrak{S}(\mathbf{w} - \mathbf{d}, \frac{\lambda}{2\mu})$ 
5:      $\mathbf{w} \leftarrow \arg \min_{\mathbf{w}} \|\mathbf{y} - \mathbf{H}\Psi'\mathbf{w}\|_2^2 + \mu\|\mathbf{w} - \mathbf{v} - \mathbf{d}\|_2^2$ 
        $= \frac{1}{\mu}(\mathbf{I} - \Psi\Psi') [\Psi\mathbf{H}'\mathbf{y} + \mu(\mathbf{v} + \mathbf{d})]$ 
        $+ \Psi(\mu\mathbf{I} + \mathbf{H}'\mathbf{H})^{-1} [\mathbf{H}'\mathbf{y} + \mu\Psi'(\mathbf{v} + \mathbf{d})]$ 
6:      $\mathbf{d} \leftarrow \mathbf{d} - (\mathbf{w} - \mathbf{v})$ 
7:   end while
8:   return  $\mathbf{x} = \Psi'\tilde{\mathbf{w}}$ 
9: end procedure

```

The synthesis prior formulation is often considered as the same problem with (31) or (36) with a different transfer function $\mathbf{H}\Psi'$, and can be solved with the same algorithms. However due to the change of transfer function, $(\mu\mathbf{I} + \Psi\mathbf{H}'\mathbf{H}\Psi')^{-1}$ is needed instead of $(\mu\mathbf{I} + \mathbf{H}'\mathbf{H})^{-1}$ which does not have any closed form solution for an arbitrary Ψ and can be tedious to compute numerically. Assuming Ψ is a tight frame satisfying $\Psi'\Psi = \mathbf{I}$, we propose to use

$$(\mu\mathbf{I} + \Psi\mathbf{H}'\mathbf{H}\Psi')^{-1} = \frac{1}{\mu}\mathbf{I} - \Psi\mathbf{H}'(\mu\mathbf{I} + \mathbf{H}\mathbf{H}')^{-1}\mathbf{H}\Psi' \quad (41)$$

$$= \frac{1}{\mu}\mathbf{I} - \Psi\left(\frac{1}{\mu}\mathbf{I} - (\mu\mathbf{I} + \mathbf{H}'\mathbf{H})^{-1}\right)\Psi' \quad (42)$$

$$= \frac{1}{\mu}\mathbf{I} - \frac{1}{\mu}\Psi\Psi' + \Psi(\mu\mathbf{I} + \mathbf{H}'\mathbf{H})^{-1}\Psi' \quad (43)$$

by making use of the matrix inversion lemma. Therefore the overall speed of the algorithm depends on the fast computation of $(\mu\mathbf{I} + \mathbf{H}'\mathbf{H})^{-1}$. The algorithm solving (21) with ADMM with Ψ being a tight frame is summarized in Algorithm 1. Note that the term $\left(\frac{1}{\mu}\mathbf{I} - \frac{1}{\mu}\Psi\Psi'\right)$ vanish when Ψ is an orthonormal transform and the analysis and synthesis prior formulations become equivalent as are the steps of the algorithms solving (31), (36) and (39). But when an overcomplete transform satisfying $\Psi'\Psi = \mathbf{I}$ is used and a fast computation of $(\mu\mathbf{I} + \mathbf{H}'\mathbf{H})^{-1}$ is available, Algorithm 1 can be much faster than synthesis prior minimization algorithms such as FISTA, or other ADMM based algorithms such as SALSA that uses $(\mu\mathbf{I} + \Psi\mathbf{H}'\mathbf{H}\Psi')^{-1}$.

B. ADMM for Analysis Prior

A more challenging problem is the analysis prior formulation in (23) which can be approached similar to synthesis prior

Algorithm 2 ADMM minimization to solve (23)

```

1: procedure ADMM_Analysis( $\mathbf{y}, \mathbf{H}, \mathbf{R}, \lambda$ )
   Minimizes  $\lambda\|\mathbf{v}\|_1 + \|\mathbf{y} - \mathbf{H}\mathbf{x}\|_2^2$  s.t.  $\mathbf{v} = \mathbf{R}\mathbf{m}, \mathbf{m} = \mathbf{x}$ 

2:   Set  $\mu_1, \mu_2 > 0, \mathbf{d}_1, \mathbf{d}_2 = 0, \mathbf{x} = \mathbf{H}'\mathbf{y}, \mathbf{m} = \mathbf{x}$ 
3:   while convergence criteria not met do
4:      $\mathbf{v} \leftarrow \arg \min_{\mathbf{v}} \lambda\|\mathbf{v}\|_1 + \mu_1\|\mathbf{v} - \mathbf{R}\mathbf{m} - \mathbf{d}_1\|_2^2$ 
        $= \mathfrak{S}(\mathbf{R}\mathbf{m} + \mathbf{d}_1, \frac{\lambda}{2\mu_1})$ 
5:      $\mathbf{m} \leftarrow \arg \min_{\mathbf{m}} \mu_1\|\mathbf{v} - \mathbf{R}\mathbf{m} - \mathbf{d}_1\|_2^2$ 
        $+ \mu_2\|\mathbf{m} - \mathbf{x} - \mathbf{d}_2\|_2^2$ 
        $= \left(\frac{\mu_2}{\mu_1}\mathbf{I} + \mathbf{R}'\mathbf{R}\right)^{-1} \left[\mathbf{R}'(\mathbf{v} - \mathbf{d}_1) + \frac{\mu_2}{\mu_1}(\mathbf{x} + \mathbf{d}_2)\right]$ 
6:      $\mathbf{x} \leftarrow \arg \min_{\mathbf{x}} \|\mathbf{y} - \mathbf{H}\mathbf{x}\|_2^2 + \mu_2\|\mathbf{m} - \mathbf{x} - \mathbf{d}_2\|_2^2$ 
        $= (\mu_2\mathbf{I} + \mathbf{H}'\mathbf{H})^{-1} [\mathbf{H}'\mathbf{y} + \mu_2(\mathbf{m} - \mathbf{d}_2)]$ 
7:      $\mathbf{d}_1 \leftarrow \mathbf{d}_1 - (\mathbf{v} - \mathbf{R}\mathbf{m})$ 
8:      $\mathbf{d}_2 \leftarrow \mathbf{d}_2 - (\mathbf{m} - \mathbf{x})$ 
9:   end while
10:  return  $\mathbf{x}$ 
11: end procedure

```

formulation to minimize

$$\tilde{\mathbf{x}} = \arg \min_{\mathbf{x}} \lambda\|\mathbf{v}\|_1 + \|\mathbf{y} - \mathbf{H}\mathbf{x}\|_2^2 \quad (44)$$

such that $\mathbf{v} = \mathbf{R}\mathbf{m}, \mathbf{m} = \mathbf{x}$

Algorithm solving (44) is presented in Algorithm 2 and can be derived the same way as Algorithm 1. The auxiliary variable \mathbf{m} in (44) essentially decouples the regularization and data fidelity as can be seen in lines 5 and 6 of Algorithm 2, and the inversions $\left(\frac{\mu_2}{\mu_1}\mathbf{I} + \mathbf{R}'\mathbf{R}\right)^{-1}$ and $(\mu_2\mathbf{I} + \mathbf{H}'\mathbf{H})^{-1}$ are guaranteed to exist.

In order for each iteration of Algorithm 2 to be fast, the terms $\left(\frac{\mu_2}{\mu_1}\mathbf{I} + \mathbf{R}'\mathbf{R}\right)^{-1}$ (line 5) and $(\mu_2\mathbf{I} + \mathbf{H}'\mathbf{H})^{-1}$ (line 6) must be calculated efficiently and if possible precisely. Closed form solution to $\left(\frac{\mu_2}{\mu_1}\mathbf{I} + \mathbf{R}'\mathbf{R}\right)^{-1}$ exist for many common \mathbf{R} and it can otherwise be calculated with few Conjugate Gradient (CG) iterations due to $\mathbf{R}'\mathbf{R}$ often having only few zero eigenvalues. Therefore the computation speed of $(\mu_2\mathbf{I} + \mathbf{H}'\mathbf{H})^{-1}$ is the main factor determining the speed of each iteration of Algorithm 2. Note that when compared to traditional ADMM approach in (31), Algorithm 2 can be much faster when $(\mu_2\mathbf{I} + \mathbf{H}'\mathbf{H})^{-1}$ can be quickly computed or has a closed form solution unlike $(\mu\mathbf{R}'\mathbf{R} + \mathbf{H}'\mathbf{H})^{-1}$, but it will be slower when $\mathbf{R}'\mathbf{R} = \mathbf{I}$ due to the extra variable separation. When compared to SALSA, the extra variable separation decoupling the data fidelity and regularization also provides advantage for regularization terms such as TV, removing the need for other iterative algorithms for TV minimization. The iterations of Algorithm 2 will also converge faster than P2 algorithm in [15] due to having fewer auxiliary variables without introducing additional computational complexity assuming

computation of $(\mu_2 \mathbf{I} + \mathbf{H}'\mathbf{H})^{-1}$ can be fast and accurate.

C. Fast Computation of $(\mu \mathbf{I} + \mathbf{H}'\mathbf{H})^{-1}$ for Cartesian Sampled Parallel MRI

Both Algorithms 1 and 2 rely on fast and efficient computation of $(\mu \mathbf{I} + \mathbf{H}'\mathbf{H})^{-1}$ especially for a large \mathbf{H} . We propose an efficient and exact calculation by making use of the separability of the transfer function among the horizontal and vertical axes in Cartesian sampled Parallel MRI. Deriving $\mathbf{H}'\mathbf{H}$ from (14) and (9) and using $\mathbf{F}'_h \mathbf{F}_h = \mathbf{I}$ we have

$$\mathbf{H}'\mathbf{H} = \begin{bmatrix} \mathbf{H}'_1 \mathbf{H}_1 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & \mathbf{H}'_{n_t} \mathbf{H}_{n_t} \end{bmatrix} \quad (45)$$

$$\mathbf{H}'_t \mathbf{H}_t = \sum_{c=1}^{n_c} \mathbf{H}'_{t,c} \mathbf{H}_{t,c} \quad (46)$$

$$= \sum_{c=1}^{n_c} \mathbf{S}'_c \tilde{\mathbf{F}}'_{v,t} \mathbf{F}'_h \mathbf{F}_h \tilde{\mathbf{F}}_{v,t} \mathbf{S}_c \quad (47)$$

$$= \sum_{c=1}^{n_c} [(\mathbf{S}_c \mathbf{S}'_c) \odot \tilde{\mathbf{F}}_{v,t} \tilde{\mathbf{F}}_{v,t}] \quad (48)$$

$$= (\tilde{\mathbf{F}}'_{v,t} \tilde{\mathbf{F}}_{v,t}) \odot \sum_{c=1}^{n_c} (\mathbf{S}_c \mathbf{S}'_c) \quad (49)$$

Note from (49) that $\mathbf{H}'\mathbf{H}$ is disjoint along the horizontal axis as well as the time axis. Therefore it is possible to have singular value decomposition (SVD) of $\mathbf{H}'\mathbf{H} = \mathbf{U}\mathbf{\Lambda}\mathbf{U}'$ as $n_h \times n_t$ separate SVDs of $n_v \times n_v$ matrices such that

$$\mathbf{U} = \begin{bmatrix} \mathbf{U}_1 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & \mathbf{U}_{n_t} \end{bmatrix}, \mathbf{U}_t = \begin{bmatrix} \mathbf{U}_{v,1,t} & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & \mathbf{U}_{v,n_h,t} \end{bmatrix}, \quad (50)$$

$$\mathbf{\Lambda} = \begin{bmatrix} \mathbf{\Lambda}_1 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & \mathbf{\Lambda}_{n_t} \end{bmatrix}, \mathbf{\Lambda}_t = \begin{bmatrix} \mathbf{\Lambda}_{v,1,t} & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & \mathbf{\Lambda}_{v,n_h,t} \end{bmatrix}, \quad (51)$$

$$\mathbf{U}_{v,i,t} \mathbf{\Lambda}_{v,i,t} \mathbf{U}'_{v,i,t} = (\tilde{\mathbf{F}}'_{v,i,t} \tilde{\mathbf{F}}_{v,i,t}) \odot \sum_{c=1}^{n_c} (\mathbf{S}_{v,i,c} \mathbf{S}'_{v,i,c}) \quad (52)$$

which is significantly easier to compute than non-separable case. Assuming we have the SVD of $\mathbf{H}'\mathbf{H} = \mathbf{U}\mathbf{\Lambda}\mathbf{U}'$ where $\mathbf{\Lambda}$ is a diagonal matrix with eigenvalues $\{e_k\}$ on main diagonal and $\mathbf{U}\mathbf{U}' = \mathbf{U}'\mathbf{U} = \mathbf{I}$,

$$(\mu \mathbf{I} + \mathbf{H}'\mathbf{H})^{-1} = (\mu \mathbf{I} + \mathbf{U}\mathbf{\Lambda}\mathbf{U}')^{-1} = \mathbf{U}(\mu \mathbf{I} + \mathbf{\Lambda})^{-1} \mathbf{U}' \quad (53)$$

$$= \mathbf{U} \bar{\mathbf{\Lambda}}_{\mu} \mathbf{U}' \quad (54)$$

with $\bar{\mathbf{\Lambda}}_{\mu}$ a diagonal matrix with the diagonal elements $\left\{ \frac{1}{e_k + \mu} \right\}$ where e_k are the eigenvalues of $\mathbf{\Lambda}$. In addition to being significantly faster, the computation of SVD and $(\mu \mathbf{I} + \mathbf{H}'\mathbf{H})^{-1}$ can easily be implemented in parallel since the equations are disjoint for each column of \mathbf{x} at time t , $\mathbf{x}_{v,i,t}$ (52).

V. EXPERIMENTAL RESULTS

Experiments are conducted on a 2D cardiac MRI dataset acquired on a 3T Siemens Trio scanner using a 32-coil matrix body array. Fully sampled data were acquired using a 128×128 matrix (FOV = 320×320 mm) and 22 temporal frames. The fully sampled data is then retrospectively undersampled by a factor of 8 using a different variable density random undersampling along vertical axis for each time point as shown in Figure 1. The data is normalized so that the reconstruction of the fully sampled data leads to images with maximum intensity of 1. The sample frames from reconstruction of the fully sampled data can be seen in Figure 2. The subsampled data is reconstructed with Algorithm 1 and Algorithm 2 using the SVD method to compute the term $(\mu \mathbf{I} + \mathbf{H}'\mathbf{H})^{-1}$ and setting temporal DFT as Ψ and temporal TV as \mathbf{R} respectively. The same λ value that results in good quality reconstruction is used for all the tested algorithms ($\lambda = 0.002$).

The reconstruction performance with formulations in (31) and (38) are also presented as P1(n) and P2 respectively where n represents the number of conjugate gradient iterations used for P1. The DFT is selected mainly for the purpose of comparing Algorithm 1 to P1, and P2 for which the analysis and synthesis prior problems become equivalent and the drawbacks of using CG steps are more apparent. SALSA algorithm with the proposed fast computation of $(\mu \mathbf{I} + \mathbf{H}'\mathbf{H})^{-1}$ is used for TV regularization only, due to Algorithm 1 and SALSA also being equivalent for orthonormal DFT regularization. The reconstruction with algorithms FISTA (for DFT) and MFISTA (for TV) are provided for comparison to non-ADMM methods. μ_i for each algorithm is selected empirically through simulations to yield for fastest convergence of each algorithm. In our experiments, selecting μ in Algorithm 1 (and μ_2 for Algorithm 2) as 0.06 and the ratio $\frac{\mu_2}{\mu_1}$ in Algorithm 2 as 0.5 or 0.25 resulted in sufficiently fast convergence. The inner iterations for MFISTA algorithm as well as for SALSA are similarly selected as 10.

All algorithms are implemented in MATLAB[®] R2010b [20] by the authors and the simulations are performed on a MacPro5,1 computer with 2.8 GHz quad-core Intel Xeon central processing unit (CPU), 6 gigabytes of memory and MacOSX 10.6.8 operating system. Apart from the simulations on the CPU, each algorithm is also accelerated on GPU with the help of JACKET[®] 2.0 toolbox for MATLAB [21] and NVidia Quadro FX 4800 graphics card with 1.5 gigabytes of memory and 192 CUDA processor cores to demonstrate how much further each algorithm can be accelerated with the help of parallel processing. JACKET[®] 2.0 toolbox provides built in functions to perform almost all operations on GPU which can be as simple as array or matrix multiplication or more complicated such as Fast Fourier Transform or SVD. It also provides the option to perform loops with independent iterations in parallel therefore enabling parallel processing of sequential tasks. All the tested algorithms are implemented in order to utilize GPU as efficient as possible, keeping the steps of the algorithms that does not benefit from GPU acceleration on CPU in order to maximize available memory in the graphics card. Each simulation is initialized with $\mathbf{x}_{initial} = \mathbf{H}'\mathbf{y}$

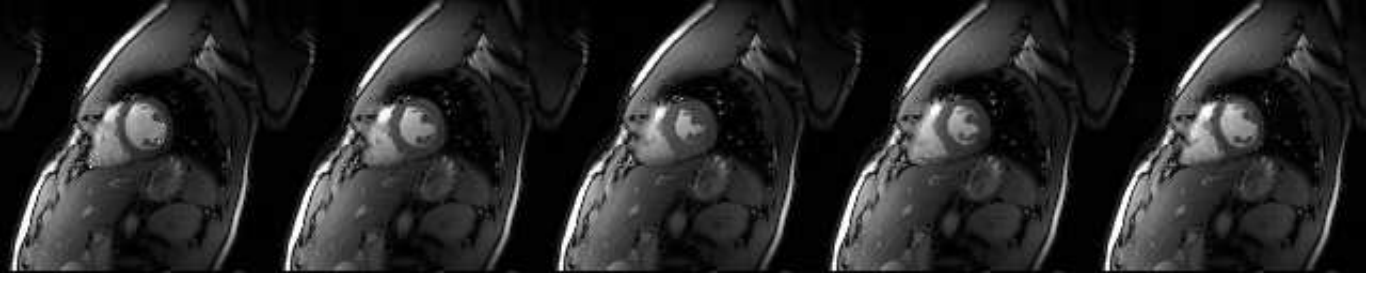
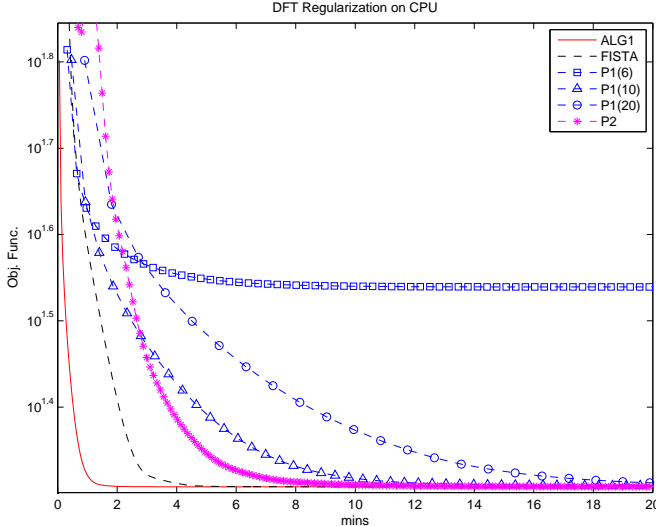
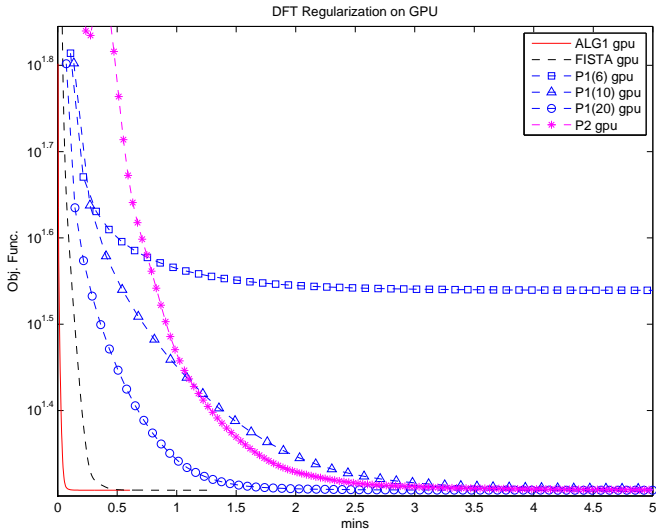


Fig. 2. Frames 1, 4, 7, 10, 12 of the cine dataset



(a) CPU



(b) GPU

Fig. 3. The objective function for each algorithm with temporal DFT regularization

and performed with 200 iterations for 10 times on minimum operating system load and the fastest among the 10 is reported for the time measure in the results.

The convergence speed of FISTA, Algorithm 1 and P1 for temporal DFT regularization can be seen in Figure 3. For an

orthonormal transform, Algorithm 1 and P1 have basically the same update on the reconstruction at each iteration, however the difference in their speed is significant mainly due to CG iterations vs SVD based computation. For a severely ill conditioned and large \mathbf{H} as in the experiment, the inaccuracy of CG iterations results in a major impact on the speed of the algorithm. Even when the algorithms are accelerated on GPU, P1 is unable to achieve the speed of CPU implementations of Algorithm 1 or FISTA as observed in Figure 5a. P2 algorithm perform better than P1 for CPU simulations whereas P1 is accelerated with GPU implementation much better than P2. This is expected since execution of each step of P2 algorithm is already fast and cannot be accelerated by GPU implementation as much as the other algorithms therefore the slow convergence of P2 algorithm becomes apparent when computational drawbacks are reduced by GPU computation.

In our study we use the ratio of reduction in the objective function as the convergence criterion which is defined as

$$\Delta(k) \triangleq \frac{J(k-1) - J(k)}{J(k)} \quad (55)$$

where $J(k)$ is the value of the minimized objective function at iteration k . The time it takes for each algorithm to reach $\Delta(k) = 0.001$ is shown in Figure 5a. It can be seen from Figure 5a that Algorithm 1 with SVD decomposition is already fastest when executed on CPU, and is accelerated multiple times more than the other algorithms when GPU is utilized.

For the analysis prior formulation with temporal TV, all the algorithms take longer time compared to DFT regularization case due to \mathbf{R} being ill conditioned as well as \mathbf{H} . The performance of the P1 algorithm can be seen to be affected by this the most since the term $(\mu \mathbf{R}'\mathbf{R} + \mathbf{H}'\mathbf{H})^{-1}$ becomes much harder to estimate. It can be seen from Figure 4 that the algorithm even fails to converge unless the number of CG iterations are kept higher than 10. Although the GPU acceleration improves the performance, the results are far from the performance of MFISTA or Algorithm 2. The speed of P2 algorithm is close to P1 for CPU simulation but falls behind during simulations with GPU similar to the case with DFT regularization. SALSA performs better than all other compared algorithms but still falls behind Algorithm 2 due to having multiple iterations of Chambolle's Algorithm for TV minimization at every step. The converged objective function value is also slightly higher than MFISTA and Algorithm 2 due to inaccuracy of the TV estimation with few iterations. The convergence speed of the algorithms can be observed in

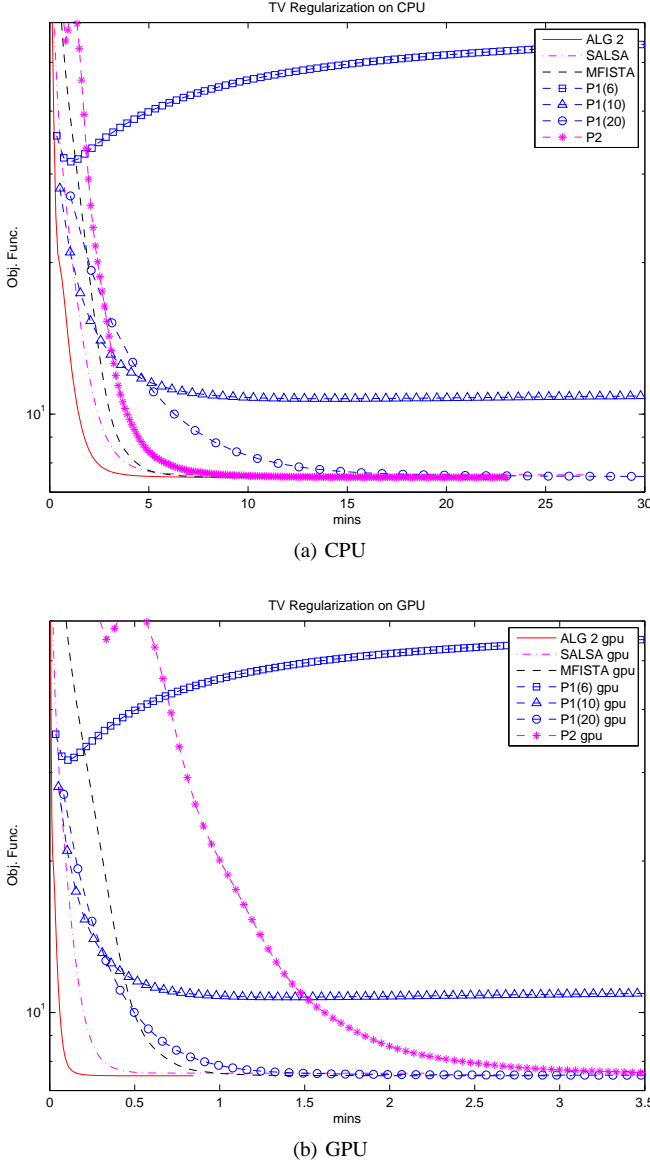


Fig. 4. The objective function for each algorithm with temporal TV regularization

Figure 5b.

It can be observed from Figures 3 and 4 that the performance of both P1 and P2 are worse than FISTA and MFISTA unlike the reported performance in [15]. The high undersampling ratio in dynamic MRI applications lead to a challenging reconstruction problem with slow convergence unless the optimization algorithms are fast and accurate, and P1 and P2 are severely affected by \mathbf{H} . Both FISTA and MFISTA have computationally simple steps and for a large \mathbf{H} the main computational complexity lies within calculation of $\mathbf{H}^H\mathbf{H}$, which can be efficiently calculated thanks to the separability shown in (49). As a result the speed of FISTA and MFISTA are improved much more significantly than P1 and P2 especially when implemented in GPU. The computational speed of these algorithms is therefore comparable to Algorithm 1 and 2 but overall speed is slower due to faster iterations of ADMM formulation.

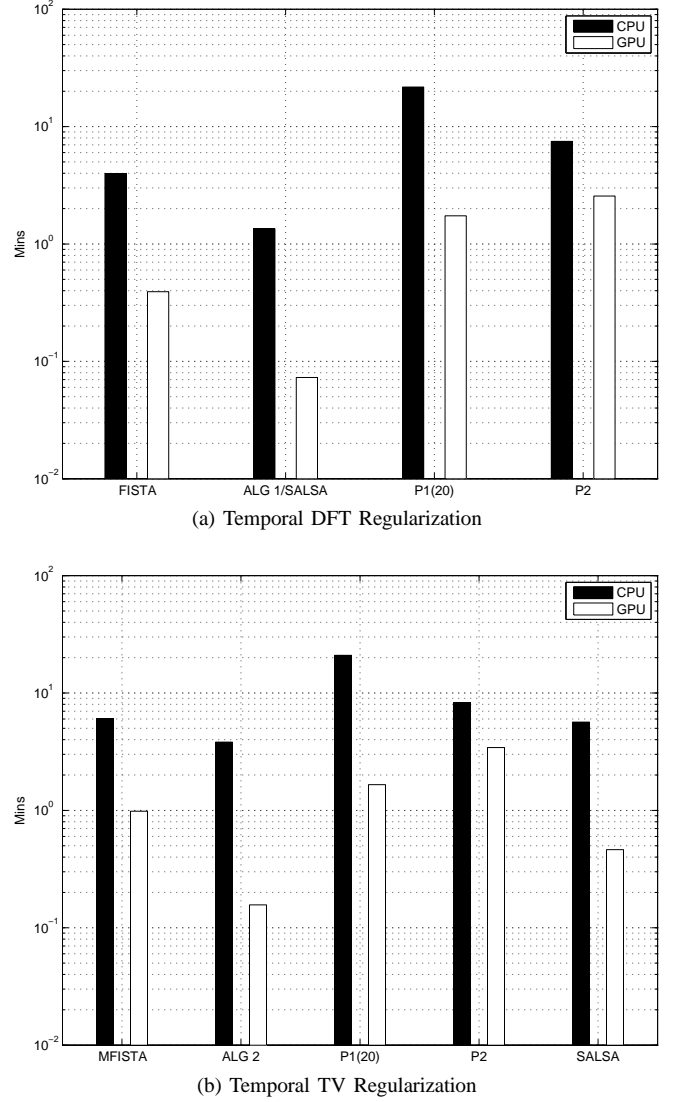


Fig. 5. Comparison of different algorithms in terms of time spent to reach $\Delta(k) = 0.001$ defined in (55)

It should be noted that although the presented algorithms are faster, their implementation require a large memory due to simultaneously storing and using the SVD of $\mathbf{H}^H\mathbf{H}$. This is not a significant drawback since systems with high memory and computational power are available in commercial configurations. The optimization can also be separated along the horizontal axis when the temporal regularization is used, in order to perform the reconstruction sequentially to reduce the memory requirement if necessary, although this would reduce the acceleration gain. The calculation of the initial SVD of $\mathbf{H}^H\mathbf{H}$ is not included in any of the presented simulation results since it can be calculated independently along the horizontal and temporal axis in parallel resulting in only a small computational overhead especially when subsampling ratio is high and $\mathbf{H}^H\mathbf{H}$ has a lot of zero eigenvalues.

VI. CONCLUSIONS

In this paper two new ADMM based algorithms are presented to solve the synthesis and analysis prior regularization

problem for compressed sensing in parallel MRI. The proposed algorithms make use of the ADMM formulation to provide faster convergence rate than other state of the algorithms such as FISTA, MFISTA, SALSA or traditional Split-Bregman Method. A computational method to improve the speed of the proposed algorithms is also presented which makes use of the separability of the transfer function in Cartesian sampled MRI, leading to faster convergence than other ADMM based methods. The performance of the proposed algorithms are shown to be significantly faster than other methods for a highly undersampled dynamic MRI application and the performance gap with the other algorithms is shown to increase when a parallel processing system is utilized such as GPUs.

As a drawback of utilizing singular value decomposition of the transfer function to gain speed, implementation of presented algorithms require more memory than other methods and this requirement would increase with larger images or volumes. Although this does not present a problem for systems with large memory, it can still be overcome if needed by reconstructing the signal sequentially along the horizontal axis. The acceleration factor can easily be multiplied with the use of more advanced GPUs and better implementation on faster languages such as C/C++.

VII. ACKNOWLEDGEMENTS

The authors would like to thank Li Feng, Daniel Kim, Ricardo Otazo and Daniel K. Sodickson from NYU Medical Center for their help and support as well as for providing the cardiac MRI dataset. We also thank Yilin Song for his assistance in performing the simulations.

REFERENCES

- [1] E. Candes and M. Wakin, "An Introduction To Compressive Sampling," *IEEE Signal Processing Magazine*, vol. 25, no. 2, pp. 21–30, Mar. 2008.
- [2] M. Lustig, D. Donoho, J. Santos, and J. Pauly, "Compressed Sensing MRI," *IEEE Signal Processing Magazine*, vol. 25, no. 2, pp. 72–82, Mar. 2008.
- [3] B. Liu, Y. M. Y. Zou, and L. Ying, "SparseSENSE: application of compressed sensing in parallel MRI," in *Information Technology and Applications in Biomedicine, 2008. ITAB 2008. International Conference on*, vol. 2, no. 3, IEEE, May 2008, pp. 127–130.
- [4] G.-H. Chen, J. Tang, and J. Hsieh, "Temporal resolution improvement using PICCS in MDCT cardiac imaging," *Medical Physics*, vol. 36, no. 6, p. 2130, 2009.
- [5] U. Gamper, P. Boesiger, and S. Kozerke, "Compressed sensing in dynamic MRI," *Magnetic resonance in medicine : official journal of the Society of Magnetic Resonance in Medicine / Society of Magnetic Resonance in Medicine*, vol. 59, no. 2, pp. 365–73, Feb. 2008.
- [6] R. Otazo, D. Kim, L. Axel, and D. K. Sodickson, "Combination of compressed sensing and parallel imaging for highly accelerated first-pass cardiac perfusion MRI," *Magnetic resonance in medicine : official journal of the Society of Magnetic Resonance in Medicine / Society of Magnetic Resonance in Medicine*, vol. 64, no. 3, pp. 767–76, Sep. 2010.
- [7] M. Usman, C. Prieto, T. Schaeffter, and P. G. Batchelor, "k-t group sparse: A method for accelerating dynamic MRI," *Magnetic resonance in medicine : official journal of the Society of Magnetic Resonance in Medicine / Society of Magnetic Resonance in Medicine*, vol. 1176, pp. 1163–1176, Mar. 2011.
- [8] M. S. Hansen, D. Atkinson, and T. S. Sorensen, "Cartesian sense and k-t sense reconstruction using commodity graphics hardware," *Magnetic Resonance in Medicine*, vol. 59, no. 3, pp. 463–468, 2008.
- [9] T. Sorensen, D. Atkinson, T. Schaeffter, and M. Hansen, "Real-time reconstruction of sensitivity encoded radial magnetic resonance imaging using a graphics processing unit," *Medical Imaging, IEEE Transactions on*, vol. 28, no. 12, pp. 1974–1985, dec. 2009.
- [10] A. Beck and M. Teboulle, "Fast gradient-based algorithms for constrained total variation image denoising and deblurring problems," *IEEE transactions on image processing : a publication of the IEEE Signal Processing Society*, vol. 18, no. 11, pp. 2419–34, Nov. 2009.
- [11] J. Bioucas-Dias and M. Figueiredo, "A new twist: Two-step iterative shrinkage/thresholding algorithms for image restoration," *Image Processing, IEEE Transactions on*, vol. 16, no. 12, pp. 2992–3004, dec. 2007.
- [12] E. C. B. P. J. E. Stephen Boyd, Neal Parikh, "Distributed optimization and statistical learning via the alternating direction method of multipliers," *Foundations and Trends in Machine Learning*, 2011.
- [13] T. Goldstein and S. Osher, "The split bregman method for l1-regularized problems," *SIAM Journal on Imaging Sciences*, vol. 2, no. 2, pp. 323–343, 2009.
- [14] M. Afonso, J. Bioucas-Dias, and M. Figueiredo, "Fast image recovery using variable splitting and constrained optimization," *Image Processing, IEEE Transactions on*, vol. 19, no. 9, pp. 2345–2356, Sep. 2010.
- [15] S. Ramani and J. Fessler, "Parallel MR Image Reconstruction Using Augmented Lagrangian Methods," *Medical Imaging, IEEE Transactions on*, vol. 30, no. 3, pp. 694–706, 2011.
- [16] H. Rauhut, K. Schnass, and P. Vandergheynst, "Compressed Sensing and Redundant Dictionaries," *IEEE Transactions on Information Theory*, vol. 54, no. 5, pp. 2210–2219, May 2008.
- [17] D. Donoho and J. Tanner, "Observed universality of phase transitions in high-dimensional geometry, with implications for modern data analysis and signal processing," *Philosophical transactions. Series A, Mathematical, physical, and engineering sciences*, vol. 367, no. 1906, pp. 4273–93, Nov. 2009.
- [18] M. Afonso, J. Bioucas-Dias, and M. Figueiredo, "An augmented Lagrangian approach to the constrained optimization formulation of imaging inverse problems," *Image Processing, IEEE Transactions on*, no. 99, pp. 1–1, 2011.
- [19] A. Chambolle, "An algorithm for total variation minimization and applications," *Journal of Mathematical Imaging and Vision*, vol. 20, pp. 89–97, 2004, 10.1023/B:JMIV.0000011325.36760.1e.
- [20] M. Inc., "http://www.mathworks.com/products/matlab/."
- [21] A. Inc., "http://www.accelereyes.com/products/jacket/."